Docket No.: 006741.P089
Express Mail No.: EV665831468US

UNITED STATES PATENT APPLICATION

FOR

CHANGE NOTIFICATION AGENT

Inventors:

Claus Gschiermeister
Gabriel Alvarez
Dominic Poetschke
Grego Rieken

Prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, LLP
12400 Wilshire Boulevard, 7th Floor
Los Angeles, California 90025
(310) 207-3800

CHANGE NOTIFICATION AGENT

BACKGROUND

[0001] This application is claiming priority of European Patent Application No. 04107056.6, filed December 29, 2004, entitled, CHANGE NOTIFICATION AGENT.

Field

[0002] The present invention relates to a frame work for administrating changes in data objects within an information technology architecture. The invention may be used for example in logistics or in a supply chain management environment.

STATE OF THE ART

[0003] In an object-oriented computer program environment comprising many data objects and many applications wherein each application processes one or more of the data objects, objects may be subject to many changes over time. Some instances of an object change, others may be newly created. These changes may be relevant for some but not always all of the applications. The respective applications may trigger post-processes dependent on the changes for implementing the changes.

[0004] To deal with this situation, currently in an object-oriented computer program environment, data objects have to implement specific logic within their update process directly. When a data object is modified or added, each application which is based on the data object is to be modified separately. Thus, update process and post-processing (on the application side) is directly coupled.

SUMMARY OF THE INVENTION

[0005] In general, in one aspect, this invention provides a computer-implemented method for administrating data objects in an information

technology architecture comprising a plurality of data objects and a plurality of applications, wherein each application processes at least one of the data objects, the plurality of data objects being subject to changes, and wherein:

entries representative of data objects are registered in a first data structure;

entries representative of applications are registered in a second data structure, each entry comprising specifying data objects whose changes are relevant for the respective application;

the method performing the following:

receiving notifications regarding registered data objects as to changes of the data objects;

upon each receipt of a notification,

getting changed data from the notifying data object;

checking, among the registered applications, as to whether the change is relevant for the applications,

notifying each application about the change if the change is relevant for the application; and

transmitting the relevant changed data to the application.

[0006] In another aspect, the invention provides a computer-implemented framework for administrating data objects in an object-oriented computer program environment comprising a plurality of data objects, a plurality of methods for processing data objects, and a plurality of computer program applications, wherein each application makes use at least one of the data objects, whereby the data objects are subject to changes, the framework comprising:

an agent for administrating changes of data objects, the agent being configured to perform the following:

registering entries representative of data objects in a first data structure;

registering entries representative of applications in a second data structure, each entry comprising specifying data objects whose changes are relevant for the respective application;

a first method to be called by a data object for notifying the agent about changes of the calling data object;

a second method to be called by the agent for getting changed data from the notifying data object;

a third method to be called by the agent for checking, among the registered applications, as to whether the change is relevant for the applications, and notifying each application about the change if the change is relevant for the application; and

a fourth method to be called by the agent for transmitting the relevant changed data to the application after notifying the application.

[0007] Further embodiments of the invention can comprise the following features.

[0008] In both aspects of the invention, there can be implemented additionally expectation of a confirmation of changes from an application after the transmitting the changed data to the application.

[0009] Furthermore, there can be implemented a triggering of a mechanism if an expected conformation is not received.

[0010] Further, there can be provided a registering entries of sub-objects, a sub-object being a set of data which is changed in dependence on a change of a key data object.

[0011] Yet further, there can be provided transmitting the relevant changed sub-object data to the application after notifying the application.

[0012] Specifying data objects whose changes are relevant for the respective application may comprise receiving a list of fields whose changes are relevant for the respective application.

[0013] In a further embodiment of both aspects of the invention there can be provided filtering out data objects whose changes are not to be communicated to an application, prior to transmitting the relevant changed data to the application.

[0014] The registering entries representative of applications may include specifying as to which changes of a data object are relevant for the application.

[0015] The entries of data objects and applications may be registered in a customization structure of an agent.

[0016] A data object may represent one of location, location-product, and transportation lane in context of a business application.

[0017] Furthermore, the invention comprises computer-readable storage media comprising program code for performing the inventive methods, when loaded into a computer system.

[0018] One of the advantages is that the present invention provides a decoupling of the update process and post-processing. Moreover, there is more flexibility in implementing additional filtering logics and further functionality (such as queue logics), and more transparency. This leads to an improvement in performance of the whole update process.

[0019] Thus, the inventive methods and framework provide an abstraction layer between a data object and an application that is "interested" in changes on these objects. "Interested" means that the application that needs to get the changes implemented. General methods and operations relating to receiving the changes while they are stored to database, filtering and passing through those events to interested applications and triggering application methods are bundled together within the framework.

[0020] In other words, there is a complete separation between the business model layer representing the data objects and the application layer. By providing such a clean separation between both of them, it is possible to add new objects to the framework and connect additional applications and still

expect the same type of behaviour from the framework. This provides a big advantage to both, the objects and the applications, because they both only have to implement a single class once and can provide/receive changes to/from all opposite instances.

[0021] Moreover, this architecture allows functional enhancements and making adjustments to the framework itself without enforcing the connected objects and applications to change their implementations if they do not want to participate to those functional enhancements. One example might be some kind of queue logic to restart a trigger to an application in case the application did not return a success message as confirmation of the last trigger.

BRIEF DESCRIPTION OF DRAWINGS

[0022] The accompanying drawings, which constitute a part of the specification, illustrate embodiments of the invention, and may serve in connection with the description to explain principles of the invention.

[0023] In the drawings:

[0024] Fig. 1 illustrates an overview of the environment in which the invention is applied;

[0025] Fig. 2 illustrates a flow diagram of the inventive method according to one embodiment of the invention;

[0026] Fig. 3 illustrates a further embodiment of the invention;

[0027] Fig. 4 illustrates a sequence diagram of the process according to the present invention; and

[0028] Fig. 5 illustrates a diagram with classes and methods.

DETAILED DESCRIPTION

[0029] The embodiments of the invention described hereinafter are denoted a change notification agent (CNA). In the following, the term "object"

is used to refer to a particular master data object (i.e., a physical Location, Location-Product, or Transportation Lane in the context of an application, for example a logistics application or a supply chain management environment.

[0030] The term B&A-Image – abbreviation for before- and after-Image – is used to refer to different Images in the process of changing an object that are communicated over the CNA.

[0031] Fig. 1 illustrates a CNA 100 which is configured to communicate, on the one hand, with a number of data objects 10, 20, 30, and on the other hand, with a number of applications 210, 220 which each works with one or more of these objects. The objects may represent locations, location-product, and transportation lane in context of a business application. The objects are subject to changes. Furthermore, new objects may be added or objects may be deleted. As soon as there is a change in the objects, there may be changes necessary in applications as well. Since not all the applications are working on the same objects, the changes on one particular object may require changes at one application but not at another one. The CNA 100 is the tool for administrating the changes on the applications side.

[0032] Fig. 2 illustrates the inventive method of administrating data objects in the information technology architecture. The method is implemented by a process running under control of a computer program. It acts as an agent. The process begins in box 10.

[0033] In box 20, entries which are representative of data objects are registered. In the following box 30, entries representative of applications are registered. In the entries, the data objects are specified whose changes are relevant for the respective application.

[0034] Then, the agent is operable to receive notifications regarding registered data objects as to changes of the data objects, box 40.

[0035] Upon each receipt of such a notification by the agent, the program goes to box 50 where changed data from the notifying data object are requested.

[0036] Then, in box 60, the agent checks among the registered applications as to whether the change is relevant for the applications.

[0037] Each application for which the change has been determined to be relevant, is notified about the change, box 70.

[0038] Box 70 is followed by a box 80 of transmitting the relevant changed data to the application.

[0039] Finally, the notification and update process ends with box 90.

[0040] As it is clear from this description, boxes 10 to 40 concern the customization of the agent, while boxes 50 to 80 are the notification operations executed during run-time of agent and applications connected thereto. Notification operations 50 to 80 are repeated for every received notification. Therefore, the customization operations can be performed independently from the notification operations 50 to 80 (and vice versa). Once the customization has been completed, the notification operations are iteratively performed.

[0041] In a second embodiment of the invention, which is illustrated in Fig. 3, box 80 as described above is followed by a box 85 of receiving a confirmation of changes from an application. The other boxes are the same as in the first embodiment.

[0042] The object-oriented program environment may comprise data objects which are changed in dependency of other data objects. Such objects are denoted as sub-objects. These sub-objects may be registered with the agent as well.

[0043] A third embodiment of the invention deals with such sub-objects. According to this embodiment, transmission of the relevant changed sub-object data to the application is performed after the box 70 notifying the application about the changes. Again, the other boxes are the same as in the first and second embodiments described above.

[0044] In case that not all the changes of an object are relevant for each of the registered applications, the agent may maintain a list of fields of the respective data object whose changes are relevant for the respective application. Then, based on this list, an operation is performed to filter out data objects whose changes are not to be communicated to an application, prior to the operation of transmitting the relevant changed data to the application.

[0045] If a data object should make use of the CNA, the data object is registered with the CNA (refer to box 20) through the customizing part provided by in the framework. Hereto, a computer screen is presented to the developer where the relevant data concerning the object can be provided to the CNA. In the same way, the applications which are intended to make use of the CNA are registered through the customizing part provided by the invention.

[0046] As to the data to be provided at registration, an entry for an object comprises:

- an ID representative of the data object;
- an ID representative of the key of the data object;
- a flag representative of activity;
- an ID representative of the key structure of the data object;
- an ID of the wrapper class.

[0047] An entry for a sub-object to an object comprises:

- an ID representative of the sub-object;
- an ID representative of the key data object;
- an ID representative of the structure of the data object;
- an ID representative of the object key object.

[0048] The sub-objects comprise object data with an additional key to the key object.

[0049] An entry for an application comprises:
an ID representative of the application;
a flag representative of activity;

an ID representative of the expected structure of notification.

[0050] The flag representative of the activity indicates whether the defined object is actually used or not. The wrapper class denotes the name of the class which the object or the application has programmed.

[0051] With the information structure the application decides implicitly what changes on which fields are relevant for an application.

[0052] With reference to Fig. 4 and 5, the process of administrating objects and their changes is described.

[0053] The sequence diagram of Fig. 4 shows the time relationship of the execution of the process *step*s described above for just one object 10 and one application 200 once they have been registered with the CNA 100. For performing the operations, corresponding methods are provided. The methods together with their classes 510-570 are displayed in Fig. 5.

[0054] The change notification from the data object to the CNA is performed in the following way:

[0055] The data object that wants to report changes creates an instance of the object specific implemented class CL_CNA_MD_XX 510 and hands over all relevant data to this instance.

[0056] Then, the object calls the method NOTIFY_CHANGE of the class CL_OBJ_ADAPTER 520 with the change data in type of the registered structure in the customizing to that object. This structure has to include a data structure named CNA_KEY_OBJECT_PREFIX. Herein, the object has to fill an exclusive indicator to the changed object, the type of change, and an indicator if the set of data is a before (B) or an after (A) Images. In case of deletion, a B-Image should be provided. In case of a new created object an A-Image should be provided. The object should be able the set a switch, that no call-back is unwanted or not necessary. In that case, CNA will not try to read additional information for sub-objects.

[0057] The method NOTIFY_CHANGE additionally creates an instance of the main CNA-class CL_CNA_AGENT 540 and hands over to the access method SET_OBJ_CHANGED its own instance and the information about what type of object it is reporting about.

[0058] NOTIFY_CHANGE addresses the workload to the CNA-method DELEGATE_WORK.

[0059] Reading out customizing GET_CUSTOMIZING of the involved object: GET_OBJ_CUSTOMIZING, and the connected applications: GET OBSERVER CUSTOMIZING.

[0060] Check if object is active in terms of CNA customizing. Call-back to objects method GET_ CHANGED_DATA in order to get all B&A-Images of the dependent sub-objects.

[0061] One the other hand, change notification from CNA to an application is performed in the following way:

[0062] All actions on the applications side of the communication are controlled by the CNA-method NOTIFY_OBSERVERS and work as described in the following operations:

[0063] In a first operation, all the fields and sub-objects that are relevant to an application should be read out. The relevant fields of the main object are registered in the applications customizing. This structure has to include CNA_KEY_OBJECT_PREFIX. The relevant sub-objects will be provided by the call-back method GET_RELEVANT_FIELDS_FOR_CHANGE of CNA-observer class 550. Beside from that, the application can decide by switches, if it is interested in newly created data and deletions.

[0064] CNA checks on the basis of the data evaluated in the first operation for each changed object if it is relevant to one of the registered applications, and copies the entry in an application specific hand over structure. The result of this operation is one table of relevant changes plus its

corresponding dependent objects for each application. The main table is again of customized structure type, the dependents to these objects adapt the customized structure from the object.

[0065] Before the changes are finally communicated to the applications, an applications method named FILTER_CHANGED_DATA is called. Here the applications can define special logics to filter out some additional objects and prevent them to be reported.

[0066] Finally, the method SET_CHANGED_DATA is called in order to trigger applications post-processes.

[0067] The present techniques can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. Apparatus of the invention can be implemented in a computer program product tangibly embodied in a machine-readable storage device for execution by a programmable processor. Method elements according to the invention can be performed by a programmable processor executing a program of instructions to perform functions of the invention by operating on the basis of input data, and by generating output data. The invention may be implemented in one or several computer programs that are executable in a programmable system, which includes at least one programmable processor coupled to receive data from, and transmit data to, a storage system, at least one input device, and at least one output device, respectively. Computer programs may be implemented in a high-level or object-oriented programming language, and/or in assembly or machine code. The language or code can be a compiled or interpreted language or code. Processors may include general and special purpose microprocessors. A processor receives instructions and data from memories, in particular from read-only memories and/or random access memories. A computer may include one or more mass storage devices for storing data; such devices may include magnetic disks, such as internal hard disks and removable disks; magneto-optical disks; and optical disks. Storage devices suitable for tangibly embodying computer program instructions and

data include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM disks. Any of the foregoing can be supplemented by or incorporated in ASICs (application-specific integrated circuits).

[0068] The computer systems or distributed computer networks as mentioned above may be used, for example, for producing goods, delivering parts for assembling products, controlling technical or economical processes, or implementing telecommunication activities.

[0069] To provide for interaction with a user, the invention can be implemented on a computer system having a display device such as a monitor or LCD screen for displaying information to the user and a keyboard and a pointing device such as a mouse or a trackball by which the user can provide input to the computer system. The computer system can be programmed to provide a graphical or text user interface through which computer programs interact with users.

[0070] A computer may include a processor, memory coupled to the processor, a hard drive controller, a video controller and an input/output controller coupled to the processor by a processor bus. The hard drive controller is coupled to a hard disk drive suitable for storing executable computer programs, including programs embodying the present technique. The I/O controller is coupled by means of an I/O bus to an I/O interface. The I/O interface receives and transmits in analogue or digital form over at least one communication link. Such a communication link may be a serial link, a parallel link, local area network, or wireless link (e.g. an RF communication link). A display is coupled to an interface, which is coupled to an I/O bus. A keyboard and pointing device are also coupled to the I/O bus. Alternatively, separate buses may be used for the keyboard pointing device and I/O interface.

[0071] Embodiments are in the scope of the following claims.